5/1

算法分析的目的是 分析算法的效率以求改进

邻接矩阵用一个一维数组存放图中所有顶点数据；用一个二维数组存放顶点间关系（边或弧）的数据。 删除一个节点，需要对比所有元素，所以是n

与算法的时间复杂度有关的是 问题规模

一个算法是求特定问题的运算序列。

B

算法是一个有穷规则的集合，其中之规则规定了一个解决某一特定类型的问题的运算序列。

C

算法是一个对任一有效输入都能够终止的图灵机。

D

一个算法，它是满足5 个特性的程序，这5个特性是：有限性、确定性、可 行性、有0个或多个输入且有1个或多个输出。

算法：能够对一定规范的输入，在有限时间内获得所要求的输出。------C选项的有限输入能够停机的图灵机符合这一定义，所以C正确。 算法所具有的特征：-----所以D正确。 有穷性，指算法必须能在执行有限个步骤之后终止-------结合下面的确切性，所以B正确。 确切性，算法的每一步骤必须有确切的定义-----------符合选项A的描述，所以A正确。 输入项，一个算法有0个或多个输入，以刻画运算对象的初始情况，所谓0个输入是指算法本身定义除了初始条件 输出项，一个算法有一个或多个输出，以反映对输入数据加工后的结果，没有输出的算法是无意义的 可行性，算法中执行的任何计算步骤都是可以被分解为基本的可执行操作步骤，即每个计算步骤都可以在有限的时间内完成

for(int i=1;i<n;i\*=3)

for(int j=i/3;j<i;j++){

Foo();

}时间复杂度是 O(n)

其实可以代入一部分数，估算一下复杂度

斐波那契数列的函数时间复杂度为 O(2^n)

最大堆中插入一条数据的时间复杂度是O(log(n))

对有n个结点、e条边且使用邻接表存储的有向图进行广度优先遍历，其算法时间复杂度是O(n+e)

若执行函数调用表达式fib(10),函数fib被调用的次数是 177

若C(n) 表示计算次数，则 C(0) = 1; C(1) = 1; C(n) = C(n-1) + C(n-2) + 1; n>=2; 故： C(0) = 1; C(1) = 1; C(2) = 1 + 1 + 1 = 3; C(3) = 3 + 1 + 1 = 5; C(4) = 5 + 3 + 1 = 9; C(5) = 9 + 5 + 1 = 15; ……

for (int i = 1, s = 0; i <= n; ++i)

{

int t = 1;

for (int j = 1; j <= i; ++j)

t = t \* j;

s = s + t;

}下列程序的时间复杂度是 O(n^2)

时间复杂度只跟循环有关，跟常数级操作无关。因为时间复杂度是取数量级的，而不是具体计算次数。所以这个程序的时间复杂度等价于： for (int i = 1; i <= n; ++i) for (int j = 1; j <= i; ++j) 这两层循环，外层执行n次，内存平均执行n/2次，时间复杂度为O(n^2)

一个优化的程序可以生成一n个元素集合的所有子集，那么该程序的时间复杂度是 O(2^n)

每个元素分为出现和不出现两种情况，时间复杂度为O(2^n)

设二叉搜索树上有n个结点，则在二叉搜索树上查找结点的平均时间复杂度为 O(log(n))

x = m;

y = 1;

while （x - y > e）

{

x = (x + y) / 2;

y = m / x;

}

print(x); 时间复杂度

算法的时间复杂度O（n）,在n比较小的时候，规律不明显。想象一下，logX,X1/2,X1/3函数的曲线，在x比较小时区别不大。但是当x比较大时差别比较明显。 所以我们在取m>1,e>0时，不妨将m取较大数，e取较小数(当m较大时e相当于0)。然后看函数内部执行。 x=m,y=1; x-y>0; 1.x=(x+y)/2=(m+1)/2 m非常大，则 x=m/2; y=m/x, x=m/2 则 y=2; 2.x=(x+y)/2=(m/2+2)/2=m/4+1 m非常大，则 x=m/4; y=m/x, x=m/4 则 y=4; 3.x=(x+y)/2=(m/4+4)/2=m/8+2 m非常大，则 x=m/8; y=m/x, x=m/8 则 y=8; ......... x=m/2n,y=2n 当x-y=m/2 n -2 n=0时 m/2 n -2 n=0 m=22n => n=(logm)/2

O(g(n))=｛f(n)|存在正常数c和n0使得所有n>=n0有: 0<=f(n)<=cg(n)}

红黑树的插入算法复杂度最坏情况是 O(log(n))

串的朴素模式匹配算法，主要思想是对主串(S)的每一个字符作为子串(T)开头，与要匹配的字符串进行匹配。主串(S)的长度为n，要匹配的子串的长度为m，那么朴素模式匹配算法的最坏时间复杂度为O((n-m＋1)\*m)

朴素算法中最坏情况是，前面的都匹配，到最后一个就不匹配的情况，S=”aaaaaaaaaaaab“ P=”aaab“

此种情况向前m-n个都比较了n次，最后第m-n+1次进行比较了n次，比较成功，所以总共比较次数为（m-n+1）\*n次

T(n) = 25\*T(n/5) + n^2 = 25\*( 25\*T(n/25) + (n/5)2 ) + n^2 = 5^4\*T(n/52) + 2\*n^2 = 5^(2k)\*T(n/5k) + k\*n^2 根据主方法，有T(n) = aT(n/b)+O(n^d), 则a=5^(2k), b=5k, d=2, a=b^d。 所以T(n)=O(n^d\*(lgn))=O(n^2(lgn))

高效的串行算法不一定是能导出高效的并行算法

B

高效的串行算法不一定隐含并行性

C

串行算法经适当的改造有些可以变化成并行算法

D

用串行方法设计和实现的并行算法未必有效

堆排序的空间复杂度是（），堆排序中构建堆（自底向上）的时间复杂度是（）O(1)，O(n)

贪心算法

单源最短路径中的Dijkstra算法

B

最小生成树的Prim算法

C

最小生成树的Kruskal算法

5/2

假设包含t个非零元素的稀疏矩阵A含有m行n列，并采用三元组顺序表压缩存储，其快速转置算法的时间复杂度为

1.初始化所有列的非零元素的个数统计为0（n）

2.统计每一列的非零元素个数（t）

3.接着求每一列第一个非零元素的首位置（n）

4.最后对每一个非零个数转置（t）。

总共时间：2\*（n+t）

于是，时间复杂度：O（n+t）

有个长度为12的无重复有序表，按折半查找法进行查找，在表内各元素等概率情况下，查找成功所需的平均比较（三元比较）的次数为（）

画出折半查找树，发现第一层有1个结点，第二层有2个结点，第三层有4个结点，第四层有5个结点，

因此查找成功所需的平均比较次数为(1\*1+2\*2+3\*4+4\*5)/12=37/12次，

具有12个关键字的有序表，折半查找的平均查找长度3.1

第一层　１个元素　１次

第二层　２个元素　２次

第三层　４个元素　３次

第四层　５个元素　４次

总共查询次数：　１×１＋２×２＋４×３＋５×４＝３７

平均查询次数为：３７/１２＝３.１

一个线性序列（30，14，40，63，22，5），假定采用散列函数Hash(key)=key%7来计算散列地址，将其散列存储在A[0~6]中，采用链地址法解决冲突。若查找每个元素的概率相同，则查找成功的平均查找长度是

4/3

30%7=2，查找次数为1，

14%7=0，查找次数为1，

40%7=5，查找次数为1，

63%7=0，查找次数为2，

22%7=1，查找次数为1，

5%7=5，查找次数为2，

平均查找长度为(1+1+1+2+1+2)/6=4/3